def input\_spec():

return (

tf.TensorSpec([None, 20, 18], tf.float32),

tf.TensorSpec([None, 1], tf.float32)

)

def model\_fn():

model = tf.keras.models.Sequential([

tf.keras.layers.InputLayer(input\_shape=(20, 18)),

tf.keras.layers.LSTM(32, activation='tanh'),

tf.keras.layers.Dense(1, activation='relu'),

])

return tff.learning.from\_keras\_model(

model,

input\_spec=input\_spec(),

loss=tf.keras.losses.MeanSquaredError(),

metrics=[RootMeanSquaredError()])

**5 local epoch**

**100 rounds**

Training time 1104.41 sec

Run 100/100OrderedDict([('root\_mean\_squared\_error', 102.85527), ('loss', 10579.206), ('num\_examples', 93655), ('num\_batches', 1464)])

OrderedDict([('eval',

OrderedDict([('root\_mean\_squared\_error', 78.92411),

('loss', 6229.0156),

('num\_examples', 100),

('num\_batches', 100)]))])

**1 local epoch**

**100 rounds**

**Training time 330.42 sec**

**Run 100/100OrderedDict([('root\_mean\_squared\_error', 108.74016), ('loss', 11824.422), ('num\_examples', 18731), ('num\_batches', 293)])**

**OrderedDict([('eval',**

**OrderedDict([('root\_mean\_squared\_error', 78.51948),**

**('loss', 6165.3086),**

**('num\_examples', 100),**

**('num\_batches', 100)]))])**

**BATCH\_SIZE = 32**

**GLOBAL\_EPOCHS = 20**

**LOCAL\_EPOCHS = 10**

**def input\_spec():**

**return (**

**tf.TensorSpec([None, 20, 18], tf.float32),**

**tf.TensorSpec([None, 1], tf.float32)**

**)**

**def model\_fn():**

**model = tf.keras.models.Sequential([**

**tf.keras.layers.InputLayer(input\_shape=(20, 18)),**

**tf.keras.layers.LSTM(32, activation='tanh'),**

**tf.keras.layers.Dense(1, activation='linear'),**

**])**

**return tff.learning.from\_keras\_model(**

**model,**

**input\_spec=input\_spec(),**

**loss=tf.keras.losses.MeanSquaredError(),**

**metrics=[RootMeanSquaredError()])**

**trainer = tff.learning.build\_federated\_averaging\_process(**

**model\_fn,**

**client\_optimizer\_fn=lambda: tf.keras.optimizers.Adam(**

**learning\_rate = 0.003**

**),**

**server\_optimizer\_fn=lambda: tf.keras.optimizers.Adam(learning\_rate = 1.0)**

**)**

**Run 20/20OrderedDict([('root\_mean\_squared\_error', 50.64427), ('loss', 2564.8423), ('num\_examples', 187310), ('num\_batches', 5854)])**

**OrderedDict([('eval',**

**OrderedDict([('root\_mean\_squared\_error', 30.740356),**

**('loss', 944.96954),**

**('num\_examples', 100),**

**('num\_batches', 100)]))])**